AMENDMENTS TO THE SPECIFICATION:

Page 1, before paragraph [0001], insert the following heading and sub-heading:

BACKGROUND

1. Technical Field

Page 1, between paragraphs [0001] and [0002], insert the following sub-heading:

2. Related Art

Page 1, between paragraphs [0002] and [0003], insert the following heading:

BRIEF SUMMARY

Page 1, paragraph [0003]:

[0003] According to one aspect of the present exemplary embodiment ~~invention~~, there is provided a computer system having a plurality of components that can be initialized ~~initialised~~, wherein each component is configured to produce status data from which the level of need for that component to be initialized ~~initialised~~ can be inferred, the status data having a predetermined level of need associable therewith and wherein at least one component is configured to: receive status data from other components; make a comparison using the status data received from respective components; in dependence on the comparison, select one or more components for initialization

~~initialisation~~; and[[,]] issue ~~initialisation~~ initialization instructions to the selected component(s).

Page 2, paragraphs [0005]-[0009]:

[0005] The components may be hardware components, or alternatively, the components may be software components (e.g., a computer program contained in a computer-readable storage medium), running on at least one (hardware) computer device. In one embodiment, the components are software components each running on a respective computer device, the ~~initialisation~~ initialization of a component causing the device on which it runs to be re-booted or otherwise initialized ~~initialised~~.

[0006] The status data received from a component may include a plurality of status values, such as the amount of free memory in a memory location, the frequency with which a processor is accessing a memory location, or other historical or current data which can be used to determine the need for ~~initialisation~~ initialization of the component.

[0007] However, to simplify the processing to be carried out by a component receiving the status data, the status data will preferably be in the form of an ~~initialisation~~ initialization parameter. This will reduce the need for a component to evaluate and/or cross reference different data originating from the same component in order to infer the need for that component to be initialized ~~initialised~~. Furthermore, if the need for a

component to be initialized ~~initialised~~ can be expressed simply in terms of a variable, such as the elapsed time since the last initialization ~~initialisation~~, the respective values of that variable for different components can be directly compared against one another.

[0008] The level of need for a component to be initialized ~~initialised~~ can be positive or negative. Thus an ~~initialisation~~ initialization parameter may indicate the importance or urgency for a component to be initialized ~~initialised~~, but an ~~initialisation~~ initialization parameter may alternatively indicate the importance for a component not to be initialized ~~initialised~~, for example if that component is carrying out an essential task.

[0009] Preferably, each component will be configured to execute an ~~initialisation~~ initialization routine when the ~~initialisation~~ initialization parameter for that component reaches a threshold value, in which case that component will behave as an initializing ~~initialising~~ component, sending a request message to other (recipient) components, the request message requesting respective ~~initialisation~~ initialization parameters from the recipient components. Hence, when the need for a component to be initialized ~~initialised~~ becomes sufficiently high, that component can at least attempt to find out the need for other components to be initialized ~~initialised~~. This allows the amount of information which a component needs to retain about the state of other components to be reduced, since a component can when required obtain such information by transmitting a request message.

Page 3, paragraph [0010]:

[0010] Furthermore, if each component can request ~~initialisation~~ initialization parameters from other components, the need for the ~~initialisation~~ initialization requirements of different components to be stored at a central location is reduced.

Page 3, paragraphs [0012]-[0014]:

[0012] The ~~initialisation~~ initialization routine carried out by an ~~initialising~~ initializing component preferably includes the further steps of: comparing the ~~initialisation~~ initialization parameters received from other components with the ~~initialisation~~ initialization parameter for the initializing ~~initialising~~ component; and, in dependence on the comparison, making a ~~self-initialisation~~ self-initialization decision. In this way, each component can take into account the need of other components to initialize ~~initialise~~ before taking a ~~self-initialisation~~ self-initialization decision.

[0013] In a preferred embodiment, each component is configured to select components for initialization ~~initialisation~~, and to issue ~~initialisation~~ initialization instructions to the selected components. Thus, in this embodiment, even if one component is unable, due to a fault or otherwise, to act as an initiating component and select components for initialization ~~initialisation~~, another component of the system will eventually be triggered to act as an initiating device (for example, because the time since the last ~~initialisation~~ initialization of that component has exceeded a threshold

value). This fault tolerance will allow the system to maintain itself efficiently, since the task of selecting which components to initialize ~~initialise~~ can in effect be distributed across the system.

[0014] The status data associable with a value of need data will preferably be predetermined such that the data arriving at a component from other components will have a level of need associable or associated therewith when it arrives. Thus, the association between status data and the need for ~~initialisation~~ initialization will preferably be predetermined such that it is possible to infer an indication of the level of need for a component to be initialized ~~initialised~~ before a comparison is made between the status data from the different components. The comparison between the status data from different components can then be used to evaluate the relative need of two or more components to be initialized ~~initialised~~, relative to one another.

Page 4, between paragraphs [0016] and [0017], insert the following heading:

BRIEF DESCRIPTION OF THE DRAWINGS

Page 4, paragraphs [0016]-[0018]:

[0016] Preferably, the status data will be able to take one or more values within a (possibly discrete) range of values. For example, the status data may take one (or more) of (at least) three values, indicative of high, medium or low ~~initialisation~~

initialization need. The status data from different components can then be at least partially ordered in dependence on (or accordance with) the relative value of the status data from at least some of the different components.

[0017] The ~~invention~~ exemplary embodiment will now be further described, by way of example, with reference to the following drawings, in which:

[0018] FIG. 1 shows schematically a computer system according to ~~the invention~~ an exemplary embodiment;

Page 4, paragraph [0022]:

[0022] FIGS. 5I-III show [[shows]] schematically three stages involved in the selection of devices for initialization ~~initialisation~~; and[[,]]

Page 4, before paragraph [0024], insert the following heading:

DETAILED DESCRIPTION OF EXEMPLARY EMBODIMENTS

Page 5, paragraphs [0025]-[0027]:

[0025] Each computer device is able to generate an ~~initialisation~~ initialization parameter that is indicative of the need (or equivalently the urgency or desirability) for the computer device 22 to be initialized ~~initialised~~. When the ~~initialisation~~ initialization parameter of a computer device 22 reaches a threshold value, that computer device

acts as an initiating device 22a, and transmits a request message 20 over the data links 14 to the other computer devices 22, which devices then act as recipient devices 22b. The request message requests from the recipient devices 22b the current values of their respective ~~initialisation~~ initialization parameters.

[0026] The initiating device 22a is configured to compare the received ~~initialisation~~ initialization parameters from the recipient devices 22b together with its own ~~initialisation~~ initialization parameter and to issue ~~initialisation~~ initialization instructions to itself and/or one or more recipient device(s) 22b in dependence on the relative values of the ~~initialisation~~ initialization parameters. Because the ~~initialisation~~ initialization parameter of a computer device 22 is a measure of the need for that device to be initialized ~~initialised~~, the computer devices 22 most needing ~~initialisation~~ initialization can be selected in ~~favour~~ favor of computer devices least requiring initialization ~~initialisation~~. Thus, those computer devices 22 which suffer disruption brought about by their ~~initialisation~~ initialization can be chosen so as to reduce the overall disruption to operation of the system 10.

[0027] In effect, the recipient devices 22b participate in an "auction", the ~~initialisation~~ initialization parameters representing "bids" for one or more available ~~initialisation~~ initialization instructions. The initiating device 22a can be viewed as an "auction~~eer~~" device, allocating ~~initialisation~~ initialization instructions to the devices which place the highest bids: that is, those devices (including the initiating device) which are

most in need of initialization ~~initialisation~~. Although in this example the bids are "sealed", in that the bid placed by one device is independent of bids placed by other devices (and different devices can "bid" at the same time), alternative embodiments are possible where a feedback routine is executed by the devices participating in the auction, such that the bid placed by one device is dependent on bids placed by another device.

Pages 5-6, paragraph [0028]:

[0028] The "auction" procedure or protocol can be ~~summarised~~ summarized with reference to FIGS. 5I-III in terms of three stages labelled I to III. In stage I, the initiating (auctioneer) device 22a announces its intention to hold an auction, by broadcasting a request message 11 to other recipient devices 22b. In stage II, devices not participating in another auction return bids 13 to the auctioneer device, whereas devices (indicated by shading) already participating in another auction with another auctioneer device (not shown) ignore the request message 11. The self-elected auctioneer device 22a also places a bid in its own auction. In stage III, after the auctioneer 22a has received and processed the bids, it informs the participants of the auction (non-shaded recipient devices 22b) of the "winning" devices which may be initialized ~~initialised~~ (or equivalently another maintenance task can be initiated). To inform the participants, the auctioneer sends a message 15 to the participants, which

message includes: the number of devices participating in the auction; the identity of the winning devices; and, the total number of devices participating in the auction.

Page 6, paragraph [0029]:

[0029] Each computer device 22 may be housed in a respective housing or casing, the data links 14 being formed by cables which extend between the housings of the respective computer devices. In such a situation, the computer devices 22 may be situated in different geographical locations. Alternatively, the computer devices 22 may be located in a common housing[[,]] and even, for example, on the same circuit board.

Page 6, paragraph [0031]:

[0031] The software component 28 includes an operating system module 40 for controlling the operation of the computer device 22, and an application program ~~programme~~ 42 for performing tasks. A monitoring stage 34 is provided for monitoring one or more status indicators indicative of the likelihood of a fault or potential fault with the computer device 22, and for producing an ~~initialisation~~ initialization parameter in dependence on the monitored status data. The monitoring stage 34 may monitor the amount of memory space in the memory 31 that is deemed available by the operating system module 40. Thus, if the application ~~programme~~ program 42 fails to release all the memory it has finished using, this "leaky ~~programme~~ program" malfunction (or "bug")

which could possibly lead to a fault can be monitored. When the amount of available memory drops below a threshold value, this can be classified by the monitoring stage as a fault or potential fault, and the monitoring stage can, in response to this fault or potential fault, issue an ~~initialisation~~ initialization parameter indicating a high need for the operating system module 40 and/or the application program 42 to be initialized ~~initialised~~.

Page 7, paragraphs [0032]-[0035]:

[0032] However, in a preferred embodiment, the monitoring stage includes a timer 35 for measuring the elapsed time since the last ~~initialisation~~ initialization (preferably the time since the last ~~initialisation~~ initialization of the operating system module 40), the ~~initialisation~~ initialization parameter being representative of this elapsed time. It is appreciated that because computer devices are likely to behave in a non-ideal way due to malfunctions such as the "leaky program ~~programme~~" malfunction, and because such malfunctions are likely to cause a deterioration in the operational capacity of a computer device over time, the elapsed time since the last ~~initialisation~~ initialization can provide a useful measure of the need for a computer device to be initialized ~~initialised~~.

[0033] The monitoring stage may take into account the tasks being performed by the device on which it is running, more important tasks having a greater weighting on

the ~~initialisation~~ initialization parameter than tasks of a lesser importance. In one embodiment, the monitoring stage monitors whether the device on which it runs is performing one or more of a predetermined number of task, and will only issue a non-zero ~~initialisation~~ initialization parameter to the if such a task is not being performed.

[0034] The ~~initialisation~~ initialization may involve a re-boot (or equivalently a re-start) of the computer device 22, or alternatively the ~~initialisation~~ initialization may simply require the application program 42 to be killed and re-started, without re-booting the entire computer device 22. As a further alternative, the ~~initialisation~~ initialization may require the amount of memory deemed in use by the application program 42 to be re-set, for example, to zero.

[0035] The ~~initialisation~~ initialization parameter generated by the monitoring stage 34 is passed to an ~~initialisation~~ initialization controller 32. The ~~initialisation~~ initialization controller 32 is configured to determine whether the ~~initialisation~~ initialization parameter (in this example, the elapsed time since the last initialization ~~initialisation~~) from the monitoring stage 34 is above a threshold value. If the ~~initialisation~~ initialization parameter is above a threshold value, the computer device 22 then acts as an initiating device 22a as indicated in FIG. 3, and executes an ~~initialisation~~ initialization routine. In the ~~initialisation~~ initialization routine, the ~~initialisation~~ initialization controller 32 of the initiating device 22a transmits at an output stage 36 one or more requests for

an ~~initialisation~~ initialization parameter to the other computer devices 22 (acting as recipient devices 22b).

Pages 7-8, paragraph [0036]:

[0036] The initiating device 22a need not know the identity of the recipient devices, nor need the initiating device know the number of existing recipient devices. Therefore, the request for ~~initialisation~~ initialization parameters from the initiating device 22a will preferably be sent in the form of a broadcast message, normally on a well-known port, and normally without being addressed to any specified recipient devices 22b. The request message will preferably contain: a time stamp indicating the time at which the message was sent; the name, address, and/or other label which identifies the initiating device 22a from which the request message originates; information indicative of a port number on which a reply with an ~~initialisation~~ initialization parameter should be sent; and a unique identifier for the request message.

Page 8, paragraphs [0037]-[0038]:

[0037] For each computer device 22, there is provided a listening stage 36 configured to monitor data arriving at that device 22, so as to detect if a request for an ~~initialisation~~ initialization parameter has been transmitted by another device 22 (the initiating device 22b), preferably on the chosen well-known port. When a request from an initiating device 22a is detected by the listening stage 36 of a recipient device 22b,

the request is passed to the ~~initialisation~~ initialization controller 32 of that device, in response to which the ~~initialisation~~ initialization controller 32 executes a response routine (steps carried out by the recipient device 22b are shown in FIG. 4).

[0038] As part of the response routine, the ~~initialisation~~ initialization controller 32 of the recipient device 22b determines whether the recipient device 22b is involved in an ~~initialisation~~ initialization procedure with another initiating device. If so, the ~~initialisation~~ initialization controller 32 does not respond to the most recent request for an ~~initialisation~~ initialization parameter. Otherwise, the ~~initialisation~~ initialization controller 32 of the recipient device 22b will access status data or an ~~initialisation~~ initialization parameter from the monitoring stage 34 of that device. The monitoring stage 34 determines if the recipient device 22b is busy performing one of a number of tasks, and if so, passes a busy signal to the ~~initialisation~~ initialization controller 32. If the recipient device 22b is not busy, the monitoring stage 34 passes to the ~~initialisation~~ initialization controller 32 an ~~initialisation~~ initialization parameter indicative of the elapsed time since the last ~~initialisation~~ initialization at the recipient device 22b. The ~~initialisation~~ initialization controller 32 of the recipient device 22b then transmits the ~~initialisation~~ initialization parameter to the initiating device 22a, unless a busy signal is received from the monitoring stage 34, in which case the ~~initialisation~~ initialization controller 32 transmits the busy signal to the initiating device 22a in the form of an ~~initialisation~~ initialization parameter which indicates the elapsed time since the last ~~initialisation~~

initialization of the recipient device 22b is zero seconds. This busy signal will allow the initiating device 22a to treat the recipient device 22b as having the lowest possible priority for initialization ~~initialisation~~.

Page 9, paragraphs [0039]-[0041]:

[0039] The listening stage 36 of the initiating device 22a captures any ~~initialisation~~ initialization parameters received from recipient devices 22b in response to the request(s), and forwards these received ~~initialisation~~ initialization parameters to the ~~initialisation~~ initialization controller 32 of the initiating device 22a. The ~~initialisation~~ initialization controller 32 then performs a comparison step using the ~~initialisation~~ initialization parameters received from recipient devices 22b and the self-generated ~~initialisation~~ initialization parameter from the monitoring stage 34 of the initiating device 22a. The comparison step involves ranking at least some of the ~~initialisation~~ initialization parameters from different devices 22 in dependence on their respective magnitudes, and selecting a proportion (such as 10%) of the ~~initialisation~~ initialization parameters having the highest values. An ~~initialisation~~ initialization instruction is then sent to the respective computer devices 22 whose ~~initialisation~~ initialization parameter falls within that proportion. (If one of those ~~initialisation~~ initialization parameters used in the comparison steps originates from the initiating device 22b, the ~~initialisation~~ initialization controller 32 of the initiating device 22b will issue a ~~self-initialisation~~

self-initialization instruction to the operating system module 40 and/or the application program 42 running on the initiating device.) [[).]] In this way, the devices 22 of the computer system 10 which most urgently require ~~initialisation~~ initialization can be selected.

[0040] The comparison step may also involve taking into account the total number of responses to the request message, as well as the number of idle devices (i.e., devices which return a non-zero value ~~initialisation~~ initialization parameter). Thus, 10% of idle devices may be selected for initialization ~~initialisation~~.

[0041] The number or percentage of devices chosen for ~~initialisation~~ initialization may be set by an administrator or the server 16, and may depend on the predicted workload of the system (the workload may be predicted by ~~analysing~~ analyzing historical patterns in the way in which the server 16 allocates tasks to the respective device, or by inspecting a log of tasks awaiting allocation by the server 16). By predicting the likely workload on the system, it is possible to estimate how many devices or what percentage of devices can be re-booted without endangering the operation of the system as a whole. In this way, the re-boot times of the devices 22 are ~~co-ordingated~~ coordinated so as to make it likely that at any one time, there will be enough working devices to handle job requests arriving at the system 10.

Pages 9-10, paragraph [0042]:

[0042] The ~~initialisation~~ initialization instructions from the initiating device 22a will preferably be sent as a common ~~initialisation~~ initialization message to the recipients devices 22b, such that the recipient devices 22b receive the same message. The ~~initialisation~~ initialization message will then contain information identifying which recipient devices 22b should be initialized ~~initialised~~, so that the ~~initialisation~~ initialization controller of [[an]] identified recipient devices 22b can act on the ~~initialisation~~ initialization message. The ~~initialisation~~ initialization message will preferably also contain information indicating the number of recipient devices 22b from which an ~~initialisation~~ initialization parameter was received and the number of recipient devices 22b identified for initialization ~~initialisation~~.

Page 10, paragraphs [0043]-[0044]:

[0043] The ~~initialisation~~ initialization instruction for a recipient device 22b will preferably instruct the recipient device 22b to initialize ~~initialise~~ within a time period. However, before instigating an initialization ~~initialisation~~, the ~~Initialisation~~ initialization controller 32 of the recipient device 22b will interrogate the monitoring stage 33 to determine if the recipient device 22b is performing any of a number of tasks (and optionally whether the tasks(s) will be continued beyond the time period allowed for in the initialization ~~initialisation~~ request). If so, the recipient device 22b is deemed to be

busy, and the ~~initialisation~~ initialization controller of the recipient device 22b ignores the initialization ~~initialisation~~ instruction. Optionally, the ~~initialisation~~ initialization controller 32 of the recipient device 22b can be configured to carry out an ~~initialisation~~ initialization in response to the ~~initialisation~~ initialization instruction if the elapsed time since the last ~~initialisation~~ initialization exceeds a threshold value, regardless of whether the recipient device 22b is busy or idle.

[0044] In FIG. 6, there is shown a UML sequence diagram illustrating the communication and relationship between software objects in the computer system (each object or process is represented as a box in FIG. 6). Each device 22 that can potentially take part in the auction runs a listening process 61 indicated by the RunListener time line. This process waits for communication from other devices. When a device has reached the threshold necessary to hold an auction, a process 63 indicated by the RunAuction time line starts. This RunAuction process 63 creates a BidManager process 65 and informs it which communication port to use. The BidManager process 65 creates a BidCollector process 67 and passes it details of the communications port on which to receive bids. The BidManager process 65 contacts the RunListener running in each ~~devices~~ device by sending a message 69. When the RunListener process 61 of a device 22 receives such a message 69, it creates a BidMaker process 71 for that device supplying device name and communications port number. The BidMaker process 71 sends a placeBid message 73 to the BidCollector

67. The BidManager also creates a BidEvaluator process 75 which calculates the winning bids i.e., which selects devices for initialization ~~initialisation~~. Winners (i.e., devices selected for initialization ~~initialisation~~) are informed via the BidCollector process and the BidMaker process. The sequence shown in FIG. 6 is preferably implemented using (or written in) the "Java" language.

Page 11, paragraphs [0045]-[0047]:

[0045] The embodiment described above is suitable for a collection of computers. Because the protocol in this embodiment does not need to know the membership details of the collection it is able to continue to work in the face of partition of the collection. As an example of this, if we consider 2 islands of computers, for example 50 in each, linked by a network connection into a single pool where the "auction protocol" of the invention is active, scheduling of maintenance or initialization ~~initialisation~~ will take place as and when required. Should the network connection fail, each island can continue scheduling maintenance within itself, until the connection is restored when the whole pool can participate again.

[0046] In another embodiment (not shown), the software components 28 run on a common computer device under the control of a common operating system. An example of this is a server running different programs for different users. In such a situation, the software components 28 will not normally include the operating system,

such that the operating system can continue to run even when one or more of the components 28 are initialized ~~initialised~~.

[0047] As can be seen from the above description, the present invention will be particularly useful in a system having components which work together as a group and which each require ~~initialisation~~ initialization at time intervals, allowing components to initialized ~~initialise~~ without excessively disturbing the group dynamics of the system.

Page 12, top of page: delete "CLAIMS" and insert the following heading:

WHAT IS CLAIMED IS: